C:\philes\the watchers: how openai, the US government, and persona built an identity surveillance machine that files reports on you to the feds

← back to philes home

the watchers: how openai, the US government, and persona built an identity surveillance machine that files reports on you to the feds

posted: **Mon Feb 16 2026 00:00:00 GMT+0000 (Coordinated Universal Time)**

ADDENDUM — february 18, 2026

we are in direct written correspondence with persona's CEO, rick song. he has been responsive and engaged in good faith.

rick has committed to answering the 18 questions in 0x14 in writing. all correspondence will be published in full as part 2 of this series. the core findings, including openai-watchlistdb.withpersona.com and its 27 months of certificate transparency history, remain unaddressed.

> ***LEGAL NOTICE***
>
> **no laws were broken.** *all findings come from passive recon using public sources - Shodan, CT logs, DNS, HTTP headers, and unauthenticated files served by the target's own web server. no systems were accessed, no credentials were used, no data was modified. retrieving publicly served files is not unauthorized access - see Van Buren v. United States (593 U.S. 374, 2021), hiQ Labs v. LinkedIn (9th Cir. 2022).*
>
> *this is protected journalism and security research under the First Amendment, ECHR Art. 10, CFAA safe harbor (DOJ Policy 2022), California Shield Law, GDPR Art. 85, and Israeli Basic Law: Human Dignity and Liberty.*
>
> *the authors are not affiliated with any government, intelligence service, or competitor of any entity named herein. no financial interest. no compensation. this research exists in the public interest and was distributed across multiple jurisdictions, dead drops, and third-party archives before publication.*
>
> *any attempt to suppress or retaliate against this publication - legal threats, DMCA abuse, employment interference, physical intimidation, or extrajudicial action - will be treated as confirmation of its findings and will trigger additional distribution. killing the messenger does not kill the message.*

> ***for the record:*** *all authors of this document are in good health, of sound mind, and have no plans to hurt themselves, disappear, or die unexpectedly. if that changes suddenly - it wasn't voluntary. this document, its evidence, and a list of names are held by multiple trusted third parties with instructions to publish everything in the event that anything happens to any of us. we mean anything.*
>
> *to Persona and OpenAI's legal teams: actually audit your supposed "FedRAMP" compliancy, and answer the questions in 0x14. that's the appropriate response. everything else is the wrong one.*

```
from:     the world
to:       openai, persona, the US government, ICE, the open internet
date:     2026-02-16
subject:  the watchers
```

---

greetz from vmfunc, MDL, Dziurwa

---

they told us the future would be convenient. sign up, verify your identity, talk to the machine. easy. frictionless. the brochure said "trust and safety." the source code said `SelfieSuspiciousEntityDetection`.

funny how that works. you hand over your passport to use a chatbot and somewhere in a datacenter in iowa, a facial recognition algorithm is checking whether you look like a politically exposed person. your selfie gets a similarity score. your name hits a watchlist. a cron job re-screens you every few weeks just to make sure you haven't become a terrorist since the last time you asked GPT to write a cover letter.

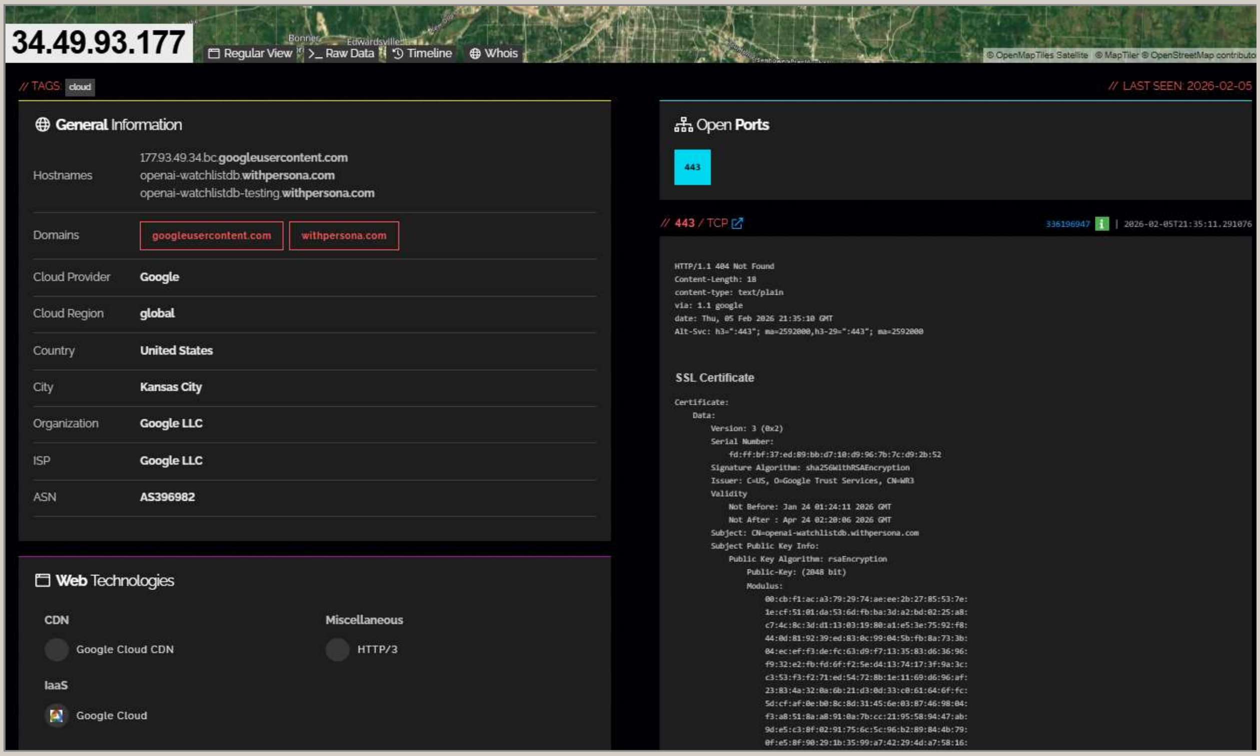
so what do you do? well, we looked. found source code on a government endpoint with the door wide open. facial recognition, watchlists, SAR filings, intelligence codenames, and much more.

oh, and we revealed the names of every single person responsible for this!!

0x00 - prologue

following the works of eva and others on ID verification bypasses, we decided to start looking into persona, yet another KYC service that uses facial recognition to verify identities. the original goal was to add a age-verification bypass to eva's existing k-id platform.

after trying to write a few exploits, vmfunc decided to browse their infra on shodan. it all started with a Shodan search. a single IP. `34.49.93.177` sitting on Google Cloud in Kansas City. one open port. one SSL certificate. two hostnames that tell a story nobody was supposed to read:


Bonner
Edwardsville
Regular View
>_ Raw Data
Timeline
Whois
© OpenMapTiles Satellite © MapTiler © OpenStreetMap contributo
443

34.49.93.177

// TAGS: cloud

// LAST SEEN: 2026-02-05

General Information

Hostnames	177.93.49.34.bc.googleusercontent.com openai-watchlistdb.withpersona.com openai-watchlistdb-testing.withpersona.com
Domains	googleusercontent.com withpersona.com
Cloud Provider	Google
Cloud Region	global
Country	United States
City	Kansas City
Organization	Google LLC
ISP	Google LLC
ASN	AS396982

Web Technologies

CDN: Google Cloud CDN

Miscellaneous: HTTP/3

IaaS: Google Cloud

Open Ports

443

// 443 / TCP 336196947 | 2026-02-05T21:35:11.291076

```
HTTP/1.1 404 Not Found
Content-Length: 18
content-type: text/plain
via: 1.1 google
date: Thu, 05 Feb 2026 21:35:10 GMT
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
```

SSL Certificate

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            fd:ff:bf:37:ed:89:bb:d7:10:d9:96:7b:7c:d9:2b:52
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=Google Trust Services, CN=WR3
        Validity
            Not Before: Jan 24 01:24:11 2026 GMT
            Not After : Apr 24 02:20:06 2026 GMT
        Subject: CN=openai-watchlistdb.withpersona.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:cb:f1:ac:a3:79:29:74:ae:ee:2b:27:85:53:7e:
                    1e:cf:51:01:da:53:6d:fb:ba:3d:a2:bd:02:25:a8:
                    c7:4c:8c:3d:d1:13:03:19:80:a1:e5:3e:75:92:f8:
                    44:0d:81:92:39:ed:83:0c:99:04:5b:fb:8a:73:3b:
                    04:ec:ef:f3:de:fc:63:d9:f7:13:35:83:d6:36:96:
                    f9:32:e2:fb:fd:6f:f2:5e:d4:13:74:17:3f:9a:3c:
                    c3:53:f3:f2:71:ed:54:72:8b:1e:11:69:d6:96:af:
                    23:83:4a:32:0a:6b:21:d3:0d:33:c0:61:64:6f:fc:
                    5d:cf:af:0e:b0:8c:8d:31:45:6e:03:87:46:98:04:
                    f3:a8:51:8a:a8:91:0a:7b:cc:21:95:58:94:47:ab:
                    9d:e5:c3:8f:02:91:75:6c:5c:96:b2:89:84:4b:79:
                    0f:e5:8f:90:29:1b:35:99:a7:42:29:4d:a7:58:16:
```

```
openai-watchlistdb.withpersona.com
openai-watchlistdb-testing.withpersona.com
```

not “openai-verify”, not “openai-kyc”, **watchlistdb**. a database. (or is it?)

it was initially meant to be a passive recon investigation, that quickly turned into a rabbit hole deep dive into how commercial AI and federal government operations work together to violate our privacy every waking second. we didn’t even have to write or perform a single exploit, the entire architecture was just on the doorstep!! 53 megabytes of unprotected source maps on a `FedRAMP` government endpoint, exposing the entire codebase of a platform that files Suspicious Activity Reports with `FinCEN`, compares your selfie to watchlist photos using facial recognition, screens you against 14 categories of adverse media from terrorism to espionage, and tags reports with codenames from active intelligence programs.

2,456 source files containing the full TypeScript codebase, every permission, every API endpoint, every compliance rule, every screening algorithm. sitting unauthenticated on the public internet. on a government platform no less.

no systems were breached. no credentials were used. every finding in this document comes from publicly accessible sources: shodan, certificate transparency logs, DNS resolution, HTTP response headers, published API documentation, public web pages, and unauthenticated JavaScript source maps served by the target’s own web server.

the infrastructure told its own story. we just listened. then we read the source code.

0x01 - the target: 34.49.93.177

```
IP:             34.49.93.177
ASN:            AS396982 (Google LLC)
provider:       Google Cloud
region:         global
city:           Kansas City, US
open ports:     443/tcp
last seen:      2026-02-05

hostnames:
  - 177.93.49.34.bc.googleusercontent.com
  - openai-watchlistdb.withpersona.com
  - openai-watchlistdb-testing.withpersona.com

SSL cert:
  subject:      CN=openai-watchlistdb.withpersona.com
  issuer:       C=US, O=Google Trust Services, CN=WR3
  valid:        Jan 24 01:24:11 2026 - Apr 24 02:20:06 2026
  SANs:         openai-watchlistdb.withpersona.com
                openai-watchlistdb-testing.withpersona.com
  serial:       FDFFBF37ED89BBD710D9967B7CD92B52

HTTP response (all paths, all methods):
  status:       404
  body:         "fault filter abort"
  headers:      via: 1.1 google
                content-type: text/plain
                Alt-Svc: h3=":443"
```

the “fault filter abort” response is an Envoy proxy fault injection filter. standard in GCP/Istio service mesh deployments. the service only routes requests matching specific internal criteria (likely mTLS client certificates, specific source IPs, or API key headers). everything else just dies at the edge.

though obviously this is not a misconfiguration.. this is just a locked-down backend service that was never meant to have a public face. the only reason we even know it exists is because of certificate transparency logs and DNS.

0x02 - dedicated infrastructure

Persona (withpersona.com) is a San Francisco-based identity verification company. their normal infrastructure runs behind Cloudflare:

```
withpersona.com          -> 162.159.141.40, 172.66.1.36  (CF)
inquiry.withpersona.com  -> 162.159.141.40, 172.66.1.36  (CF)
app.withpersona.com      -> 162.159.141.40, 172.66.1.36  (CF)
api.withpersona.com      -> 162.159.141.40, 172.66.1.36  (CF)
```

they also run a wildcard DNS record, `*.withpersona.com` points to Cloudflare (`cloudflare.withpersona.com.cdn.cloudflare.net`). we confirmed this by resolving completely fabricated subdomains:

```
totallynonexistent12345.withpersona.com -> 162.159.141.40  (CF)
asdflkjhasdf.withpersona.com            -> 162.159.141.40  (CF)
```

HOWEVER, here’s where it gets interesting. OpenAI’s watchlist service breaks out of this wildcard:

```
openai-watchlistdb.withpersona.com         -> 34.49.93.177  (GCP)
openai-watchlistdb-testing.withpersona.com -> 34.49.93.177  (GCP)
```

a dedicated Google Cloud instance, which isn’t even behind Cloudflare, nor on Persona’s shared infrastructure. seemingly purpose-built and isolated.

you would never do this for a simple “check this name against a list” API call, you do this when the data requires compartmentalization. when the compliance requirements for the data you’re collecting, demand that level of isolation. when the damage of a breach is bad enough to warrant dedicated infrastructure.

0x03 - certificate transparency timeline

CT logs tell us exactly when this service went live and how it evolved.

every certificate issuance is a public record.

date first seen	issuer	notes
2023-11-16	GTS 1D4	**SERVICE GOES LIVE**
2024-01-13	GTS 1D4	routine rotation
2024-02-28	GTS 1D4	testing gets own cert
2024-03-04	GTS 1D4	testing merged into prod cert
2024-05-02	GTS 1D4	routine rotation
2024-06-29	GTS 1D4	routine rotation
2024-08-27	GTS 1D4	routine rotation

date first seen	issuer	notes
2024-10-24	GTS 1D4	routine rotation
2024-12-21	GTS 1D4	routine rotation
2025-02-18	WR3	issuer changes
2025-04-11	WR3	routine rotation
2025-06-08	WR3	routine rotation
2025-08-05	WR3	routine rotation
2025-10-03	WR3	routine rotation
2025-11-30	WR3	routine rotation
2025-12-03	WR3	routine rotation
2026-01-24	WR3	**CURRENT CERT**

november 2023. this service has been running for over two years.

OpenAI didn't announce "Verified Organization" requirements until mid-2025. they didn't publicly require ID verification for advanced model access until GPT-5. but the watchlist screening infrastructure was operational 18 months before any of that was disclosed.

we can pinpoint when they started considering going "public" with the collaboration.

`https://withpersona.com/customers/openai` exists since September 17th, 2024, likewise, OpenAI's Privacy Policy update started including the following passage since their November 4th, 2024 update as well.

> *"Other Information You Provide: We collect other information that you provide to us, such as when you participate in our events or surveys, or when you provide us or a vendor operating on our behalf with information to establish your identity or age (collectively, "Other Information You Provide")."*

the excuses used in the public post are classical, though instead of using children as the scapegoat for invading our privacy, this time it was "[…] To offer safe AGI, we need to make sure bad people aren't using our services […].

only… that they quickly used this opportunity to go from comparing users against a single federal watchlist, to creating the watchlist of all users themselves.

in fact, this is nothing new, OpenAI Forum User OnceAndTwice had mentioned this already back in June last year.

0x04 - what the API reveals

Persona's API documentation (docs.withpersona.com) is public. when a customer like OpenAI runs a government ID verification, the API returns a complete identity dossier:

```
personal identity:
  - full legal name (including native script)
  - date of birth, place of birth
  - nationality, sex, height

address:
  - street, city, state, postal code, country

government document:
  - document type and number
  - issuing authority
  - issue and expiration dates
  - visa status
  - vehicle class/endorsements/restrictions

media:
  - FRONT PHOTO of ID document (URL)
  - BACK PHOTO of ID document (URL)
  - SELFIE PHOTO (URL + byte size)
  - VIDEO of identity capture (URL)

metadata:
  - entity confidence score
  - all verification check results with pass/fail reasons
  - capture method used
  - timestamps (created, submitted, completed, redacted)
```

Persona's own case study states that OpenAI "screens millions monthly" and "automatically screens over 99% of users behind the scenes in seconds."

behind the scenes. in seconds. millions. with customizable filters ranging from simple partial name matches to advanced facial recognition algorithms.

again, none of this is even a secret, its "hidden" in plain sight.

0x05 - the government platform: withpersona-gov.com

while investigating the watchlistdb infrastructure, we discovered a parallel deployment.

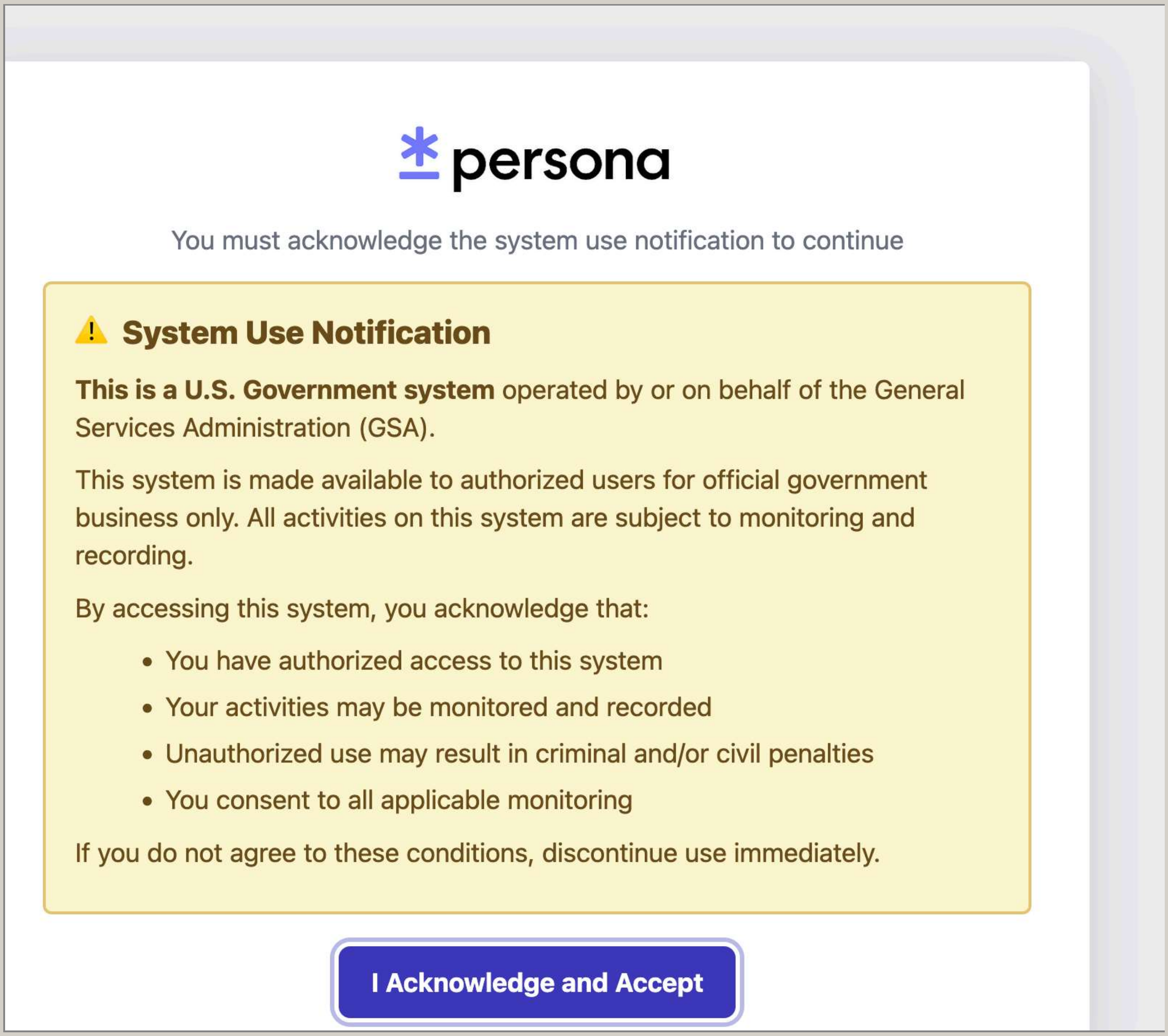
```
IP:            34.27.15.233
ASN:           AS396982 (Google LLC)
provider:      Google Cloud, us-central1
city:          Council Bluffs, US (Google's Iowa datacenter)
open ports:    80 (redirect), 443
cert:          CN=*.withpersona-gov.com (wildcard)
```

```
tech:           Caddy web server, Go, Google Cloud CDN

live subdomains:
  withpersona-gov.com           34.27.15.233   marketing
  app.withpersona-gov.com       34.27.15.233   Persona dashboard
  inquiry.withpersona-gov.com   34.27.15.233   verification flow
  admin.withpersona-gov.com     34.27.15.233   admin panel
  login-gov.withpersona-gov.com 3.15.167.135   Okta SSO (AWS!)
  app.trust.withpersona-gov.com 104.21.47.3    Cloudflare Access
  trust.withpersona-gov.com     104.21.47.3    Cloudflare Access
```

Persona achieved FedRAMP Authorized status at the Low Impact level on October 7, 2025. FedRAMP Ready at Moderate Impact. this is the deployment that serves federal agencies.

the login page at login-gov.withpersona-gov.com serves an Okta end-user dashboard. the HTML source reveals the Okta tenant: `personaforgov-admin.okta.com`



You must acknowledge the system use notification to continue

⚠ **System Use Notification**

This is a U.S. Government system operated by or on behalf of the General Services Administration (GSA).

This system is made available to authorized users for official government business only. All activities on this system are subject to monitoring and recording.

By accessing this system, you acknowledge that:

- You have authorized access to this system
- Your activities may be monitored and recorded
- Unauthorized use may result in criminal and/or civil penalties
- You consent to all applicable monitoring

If you do not agree to these conditions, discontinue use immediately.

the trust portal sits behind Cloudflare Access and the CF-Access-Domain header reveals `api.trust.withpersona-gov.com` with the application name “**fedramp-private-backend-api**”.

0x06 - the CSP header leak

the Content-Security-Policy header from withpersona-gov.com leaked their entire vendor and integration stack. every domain the government platform is allowed to communicate with:

api.openai.com - the government identity platform connects to OpenAI's API. we'll get to what this actually is in 0x0C.

FingerprintJS (`*.fpapi.io`, `*.api.fpjs.io`, `*.fptls.com`) - browser and device fingerprinting on a government identity verification platform.

Microblink (`baltazar.microblink.com`, `ping.microblink.com`) - document scanning SDK. reads and extracts data from government IDs client-side.

Sentry (`o175220.ingest.sentry.io`) - error tracking with organization ID 175220 leaked. what PII ends up in stack traces from a government identity platform?

Amplitude (`api.amplitude.com`) + **Pendo** (`app.pendo.io`) - product analytics and user behavior tracking on a government identity verification platform.

Datadog RUM (`browser-intake-datadoghq.com`) - real-time user monitoring. every click, every page load - on a FedRAMP platform processing PII and biometrics.

MX/MoneyDesktop (`int-widgets.moneydesktop.com`) - financial data widgets. financial identity verification capabilities on the government platform.

full environment map exposed:

```
production:       withpersona.com / withpersona-gov.com
staging:          withpersona-staging.com
development:      withpersona-development.com
sandbox:          withpersona-sandbox.com
staging-sandbox:  withpersona-staging-sandbox.com
```

0x07 - the ONYX deployment

on february 4, 2026 - twelve days before this document was written, a new subdomain appeared in certificate transparency logs:

```
onyx.withpersona-gov.com
```

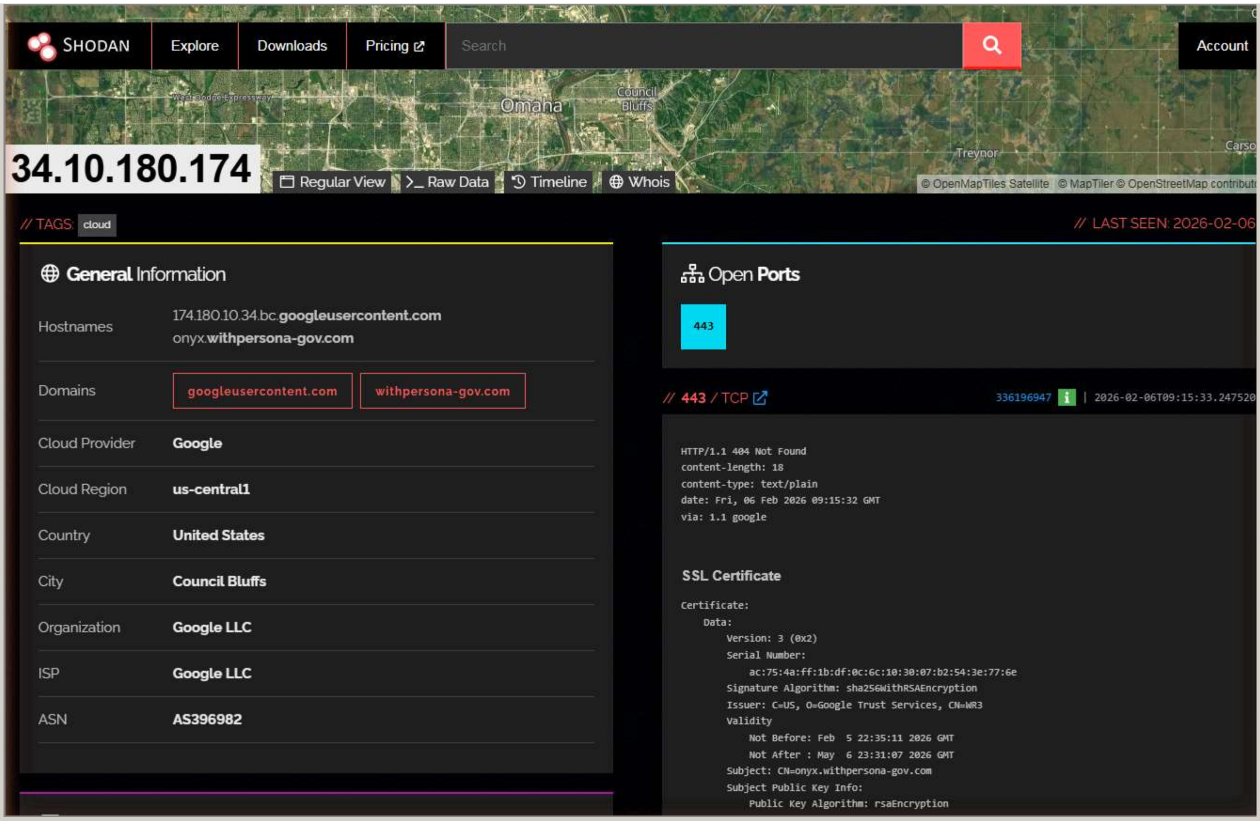
its own dedicated Google Cloud instance. its own wildcard certificate. its own Kubernetes namespace. and its name matches ICE's $4.2 million AI surveillance tool: Fivecast ONYX.

```
CT LOG TIMELINE:
  2026-02-04  first cert issued for onyx.withpersona-gov.com
  2026-02-05  second cert issued (rotation or reissue)

INFRASTRUCTURE:
  onyx.withpersona-gov.com            34.10.180.174  (GCP)
  app.onyx.withpersona-gov.com        34.10.180.174  (GCP)
  admin.onyx.withpersona-gov.com      34.10.180.174  (GCP)
  inquiry.onyx.withpersona-gov.com    34.10.180.174  (GCP)

  cert: CN=onyx.withpersona-gov.com
        SANs: onyx.withpersona-gov.com, *.onyx.withpersona-gov.com
        issuer: Google Trust Services, WR3

  server: istio-envoy (Istio service mesh)
  k8s namespace: persona-onyx
  k8s pods: web-5785f4b8f-jgm7m, web-5785f4b8f-qjg8f
  k8s service: web.persona-onyx.svc.cluster.local:3000
  git commit: 8d454ac0dc48b2f4ae7addefa22e746079c30089
  admin login: Okta-only SSO (internal staff)
  server-timing: Flipper feature flags leaked
  CSP includes: api.openai.com
  dashboard loads: faceapi.js (facial recognition)
  security.txt: Bugcrowd program, EXPIRED 2025-11-01
```


SHODAN
Explore
Downloads
Pricing
Search
Account
West Dodge Expressway
Omaha
Council Bluffs
Treynor
Carso
Regular View
>_ Raw Data
Timeline
Whois
© OpenMapTiles Satellite © MapTiler © OpenStreetMap contribut

34.10.180.174

// TAGS: cloud

// LAST SEEN: 2026-02-06

General Information

Hostnames	174.180.10.34.bc.**googleusercontent.com** onyx.**withpersona-gov.com**
Domains	googleusercontent.com withpersona-gov.com
Cloud Provider	**Google**
Cloud Region	**us-central1**
Country	**United States**
City	**Council Bluffs**
Organization	**Google LLC**
ISP	**Google LLC**
ASN	**AS396982**

Open **Ports**

443

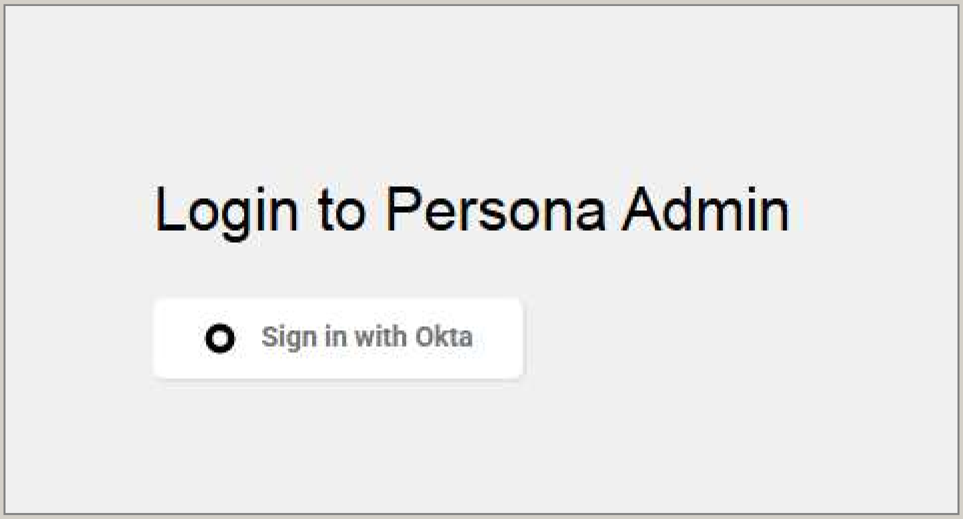
// 443 / TCP

336196947 | 2026-02-06T09:15:33.247520

```
HTTP/1.1 404 Not Found
content-length: 18
content-type: text/plain
date: Fri, 06 Feb 2026 09:15:32 GMT
via: 1.1 google
```

SSL Certificate

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            ac:75:4a:ff:1b:df:0c:6c:10:30:07:b2:54:3e:77:6e
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=Google Trust Services, CN=WR3
        Validity
            Not Before: Feb  5 22:35:11 2026 GMT
            Not After : May  6 23:31:07 2026 GMT
        Subject: CN=onyx.withpersona-gov.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
```


Login to Persona Admin
Sign in with Okta

what is Fivecast ONYX? an AI-powered surveillance platform purchased by ICE for $4.2 million and CBP for additional license costs. according to Fivecast’s own documentation and EFF’s reporting, they do automated collection of multimedia data from social media and dark web, build “digital footprints” from biographical data, tracks shifts in sentiment and emotion, assigns risk scores, searches across 300+ platforms and 28+ billion data points, identifies people with “violent tendencies”

hmm.. sounds a bit dystopian, doesn’t it? it’s not just a surveillance tool, it’s a tool that can be used for good or evil. it’s a tool that can be used to protect citizens, but it can also be used to oppress them :))

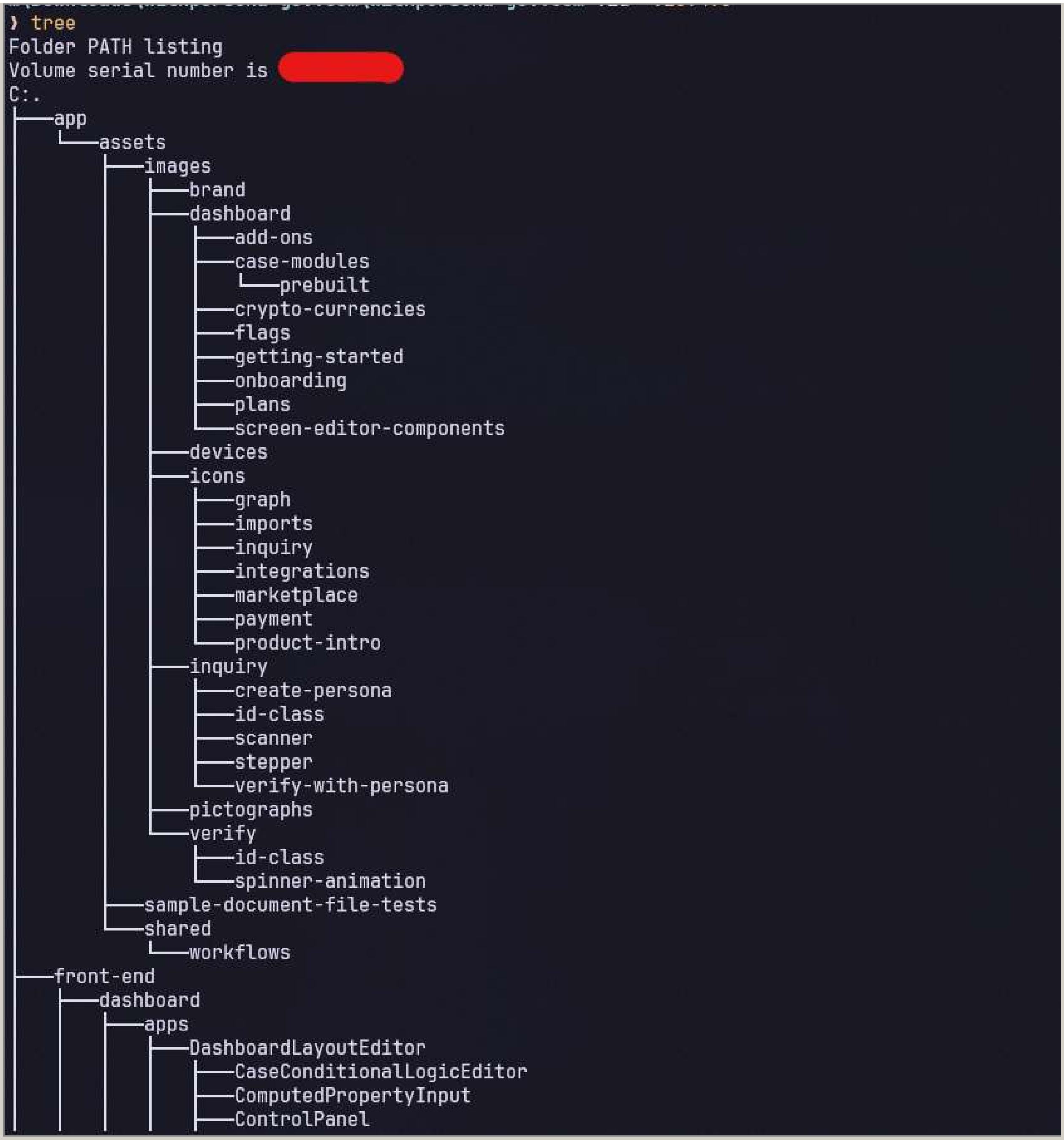
what we need to be clear about: the source code from this deployment (which we obtained in full from publicly accessible source maps, we will explain everything down below, don’t worry) contains **zero direct references** to Fivecast, ICE, immigration enforcement, or social media surveillance. no immigration workflows, no deportation tracking, no social media scraping. the code is a KYC/AML compliance platform. the ONYX name match could be coincidence, a different product entirely, or an internal codename.

or… it couldnt! they could use codenames, other platforms, what do we know.

the infrastructure correlation is real though, but the code doesn’t confirm the connection. however, what the code DOES show is significant enough on its own.

0x08 - the source maps: 53 megabytes of naked code

on the ONYX government dashboard login page (`app.onyx.withpersona-gov.com/dashboard/login`), the `/vite-dev/` asset path serves JavaScript source maps without authentication. not just minified bundles but the full, original TypeScript source code.



```
❯ tree
Folder PATH listing
Volume serial number is [redacted]
C:.
├───app
│   └───assets
│       ├───images
│       │   ├───brand
│       │   ├───dashboard
│       │   │   ├───add-ons
│       │   │   ├───case-modules
│       │   │   │   └───prebuilt
│       │   │   ├───crypto-currencies
│       │   │   ├───flags
│       │   │   ├───getting-started
│       │   │   ├───onboarding
│       │   │   ├───plans
│       │   │   └───screen-editor-components
│       │   ├───devices
│       │   ├───icons
│       │   │   ├───graph
│       │   │   ├───imports
│       │   │   ├───inquiry
│       │   │   ├───integrations
│       │   │   ├───marketplace
│       │   │   ├───payment
│       │   │   └───product-intro
│       │   ├───inquiry
│       │   │   ├───create-persona
│       │   │   ├───id-class
│       │   │   ├───scanner
│       │   │   ├───stepper
│       │   │   └───verify-with-persona
│       │   ├───pictographs
│       │   └───verify
│       │       ├───id-class
│       │       └───spinner-animation
│       ├───sample-document-file-tests
│       └───shared
│           └───workflows
├───front-end
│   ├───dashboard
│   │   ├───apps
│   │   │   ├───DashboardLayoutEditor
│   │   │   │   ├───CaseConditionalLogicEditor
│   │   │   │   ├───ComputedPropertyInput
│   │   │   │   ├───ControlPanel
```

how does this happen? vite (their build tool) generates source maps during compilation, those are `.js.map` files that contain a `sourcesContent` array with the original, unminified TypeScript embedded as JSON strings. in development, vite serves these from a `/vite-dev/` or `/@vite/` path so browser devtools can show u the real source when debugging.

vite's default config sets `build.sourcemap` to `false` for production builds… but… if someone explicitly enables it (or copies a dev config into prod) (or vibecodes their app), the `.map` files get baked into the output bundle and served alongside the minified JS. your browser already knows to look for them by default, every minified `.js` file ends with a `//# sourceMappingURL=<filename>.js.map` comment that tells devtools where to fetch the map.

on a normal deployment this is just a bad practice. on a FedRAMP-authorized government endpoint it’s **CATASTROPHIC**. the source maps don’t just contain variable names and line numbers, they contain the **entire original source** via `sourcesContent`. you can `JSON.parse()` the map file, iterate `sourcesContent`, and you have the full project tree reconstructed on disk. that’s what we did. no decompilation, no reverse engineering, no leet skills needed.

the `/vite-dev/` path specifically suggests this wasn’t even a “whoops we left sourcemaps on” situation… that path is Vite’s **development server** asset prefix. either someone deployed a dev build to production, or their Docker/k8s config is pulling from a dev stage of their CI pipeline. on a platform that went through FedRAMP security assessment. the auditors either didn’t check static assets or didn’t know what a source map was…

eiiiiither way… 53 megabytes of typescript sitting right there, for anyone with a browser.

```
17 source map files, 53 MB total:
  dashboard-BgoO_aj5.js.map        18.7 MB  (main dashboard)
  vendor-CAtBRHF0.js.map           23.8 MB  (all vendor libs)
  faceapi-CCSM7NPL.js.map           2.8 MB  (facial recognition)
  assets-dashboard-BjWtHFfU.js.map   1.4 MB  (UI components)
  sentry-CrHAK25X.js.map             1.3 MB  (error reporting)
  pdfjs-lNIjl8-7.js.map              1.0 MB  (PDF rendering)
  lottie-DtEgpYuH.js.map             1.0 MB  (animations)
  rsuite-CyQr2m5H.js.map             0.8 MB  (UI framework)
  handlebars-DPCouO2C.js.map          0.7 MB  (templating)
  redux-form-0Q1f7bM3.js.map          0.6 MB  (form state)
  assets-icons-DrRTTzXk.js.map        0.6 MB  (icon assets)
  luxon-Cv8PWZK8.js.map               0.4 MB  (date/time)
  assets-BE5pxuFi.js.map              0.3 MB  (shared assets)
  react-select-DqPrgoT_.js.map        0.3 MB  (select inputs)
  amplitude-js-b01lH43V.js.map        0.2 MB  (analytics)
  inquiry-fog-CmsaQYHz.js.map         55 KB   (inquiry workflow)
  user-authentication-BUgfjJoc.js.map 14 KB   (auth module)

extracted: 2,456 total source files
directories: front-end/ (2,056 files), app/ (400 files)
file types: TypeScript (.ts/.tsx), CSS, JSON, SVG
```

a FedRAMP-authorized government platform serving unminified source maps. this is the entire Persona dashboard codebase. every internal model, every API call, every permission check, every workflow. let’s see what it says, shall we?

before you ask: no, we can’t give you the zip. we know. we want to. believe us, we *really* want to. but the code is still Persona’s copyrighted property regardless of how monumentally they fumbled serving it to the entire internet. fair use covers the snippets and references in this phile (commentary, criticism, journalism, public interest) but shipping 2,456 files wholesale is…

different. trade secret law is even messier… courts have ruled that redistributing accidentally-exposed proprietary data can still count as misappropriation even when the owner left it sitting on a public endpoint like an idiot.

so instead: every code reference in this document cites the exact file path, function name, and relevant lines. if you want to verify, the source maps were publicly served at `app.onyx.withpersona-gov.com/vite-dev/`

whether they still are by the time you read this is between Persona and their DevOps team. we archived everything before publication. the receipts exist. they just can’t live here.

0x09 - SAR: filing suspicious activity reports directly to FinCEN

source files:

- `dashboard/views/DashboardSARShowView/DashboardSARShowView.tsx`
- `dashboard/views/DashboardSARShowView/components/SARInstructionsCard.tsx`
- `dashboard/models/filing.ts`

the platform has a full SAR module for filing directly with FinCEN (Financial Crimes Enforcement Network, US Treasury). it’s not a third-party integration or an export. they literally have a “Send to FinCEN” button.

code evidence:

```
// DashboardSARShowView.tsx - the SAR management dashboard
const handleAutofillSAR = ...   // autofill from case data
const handleValidateSAR = ...   // validate against FinCEN XML schema
const handleFileSAR = ...       // file electronically
const handleExportFincenPDF = ... // export FinCEN PDF
const handleCloneSAR = ...      // clone SAR across cases
const handleArchiveSAR = ...    // archive completed SARs

// environment mapping
{ 'environment/production': 'production', 'environment/sandbox': 'sandbox' }
```

```
// filing.ts - SAR status lifecycle
export enum FincenStatus {
  Open = 'open',
  Pending = 'pending',
  Filed = 'filed',  // deprecated
  FiledManually = 'filed_manually',
  FiledElectronically = 'filed_electronically',
  Accepted = 'accepted',
  AcceptedWithWarnings = 'accepted_with_warnings',
  Processed = 'processed',
  ProcessedWithWarnings = 'processed_with_warnings',
  Rejected = 'rejected',
  Archived = 'archived',
}
```

the instructions card links directly to https://www.fincen.gov/frequently-asked-questions-regarding-fincen-suspicious-activity-report-sar and labels it “BSA’s Support Center” (Bank Secrecy Act).

SAR permissions from the codebase:

```
Permission.SARCreate    Permission.SARList
Permission.SARView      Permission.SARWrite
Permission.SARFile      Permission.SARExport
Permission.SARConfigurationView
Permission.SARConfigurationWrite
```

government agencies using this platform can flag individuals and generate FinCEN filings, Suspicious Activity Reports sent directly to the US Treasury’s Financial Crimes Enforcement Network. the code handles the full lifecycle from creation to government acceptance or rejection.

on the other hand, there is also the possibility to file and request with a list of “Type of financial institution(s)” found under a typo’d file: `SAROganizationConfigurationSchema.ts`.

```
typeOfFinancialInstitution: {
  type: 'object',
  properties: {
    organizationTypeId: {
      type: 'number',
      title: 'Type of financial institution',
      oneOf: [
        { const: 1, title: 'Casino/Card Club' },
        { const: 2, title: 'Depository Institution' },
        { const: 3, title: 'Insurance company' },
        { const: 4, title: 'MSB (Money Service Business)' },
        { const: 5, title: 'Securities/Futures' },
        { const: 11, title: 'Loan or Finance Company' },
        { const: 12, title: 'Housing GSE (Government Sponsored Enterprise)' },
        { const: 999, title: 'Other' },
      ],
    },
  },
};
```

and the same applies to different federal regulators, too:

```
properties: {
  primaryRegulatorTypeCodeDepository: {
    title: 'Primary federal regulator',
    type: 'number',
    oneOf: [
      { const: 2, title: 'FDIC (Federal Deposit Insurance Corporation)' },
      { const: 13, title: 'FHFA (Federal Housing Finance Agency)' },
      { const: 7, title: 'IRS (Internal Revenue Service)' },
      { const: 3, title: 'NCUA (National Credit Union Administration)' },
    ],
  },
},
```

0x0A - STR: filing to FINTRAC with intelligence program tags

source files:

- `dashboard/views/DashboardFilingShowView/components/STRFormSchema.tsx` (~4000 lines)
- `dashboard/lib/filing/strs/customValidate.ts`
- `dashboard/models/filing.ts`

alongside US FinCEN, the platform files STRs (Suspicious Transaction Reports) with FINTRAC (Financial Transactions and Reports Analysis Centre of Canada). the STR form schema maps 1:1 to FINTRAC's reporting format

```
// suspicion types
suspicionTypeCode:
  1 = Money laundering
  2 = Terrorist financing
  3 = Money laundering and terrorist financing
  4 = Sanctions evasion
  5 = Money laundering and sanctions evasion
  6 = Terrorist financing and sanctions evasion
  7 = M.Launder/Terr. Finance/Sanction Evasion (all three)
```

and then, because it had to get worse, there's this:

```
// FINTRAC public-private partnership intelligence programs
publicPrivatePartnershipProjectNameCodes:
  { const: 1, title: 'Project ANTON' }
  { const: 2, title: 'Project ATHENA' }
  { const: 3, title: 'Project CHAMELEON' }
  { const: 5, title: 'Project GUARDIAN' }
  { const: 6, title: 'Project LEGION' }
  { const: 7, title: 'Project PROTECT' }
  { const: 8, title: 'Project SHADOW' }
```

these are real FINTRAC public-private partnership intelligence programs. the form lets filers tag their STR as related to specific intelligence operations by name… and they're hardcoded in the dropdown…

```
// PEP indicator on the STR
politicallyExposedPersonIncludedIndicator: boolean
// "Does this report include information about an individual who
//  you have determined to be a politically exposed person (PEP)?"
```

the e-filing submission form warns: `"This action will sign and lock the filing preventing any further edits. This action can not be undone."` it selects stored government credentials and submits directly to FINTRAC. filing credentials are stored per jurisdiction, it's username/password for FinCEN, and client-id/client-secret for FINTRAC.

cross-border financial intelligence from a single dashboard…

0x0B - face lists: biometric databases with 3-year retention

source files:

- `dashboard/components-lib/AsyncSelfie/AsyncSelfie.tsx`
- `dashboard/views/DashboardListsView/AddListModal.tsx`
- `dashboard/lib/constants/list.ts`

the platform maintains 13 types of tracking lists. from `list.ts`:

```
const LIST_TYPE_TO_LIST_ITEM_TYPE = {
  ListGovernmentIdNumber  // government ID numbers
  ListIpAddress           // IP addresses
  ListName                // names (with fuzzy matching option)
  ListPhoneNumber         // phone numbers
  ListEmailAddress        // email addresses
  ListGeolocation         // geolocations
  ListBrowserFingerprint  // browser fingerprints
  ListDeviceFingerprint   // device fingerprints
  ListFace                // FACES
  ListCountry             // countries
  ListField               // inquiry fields
  ListString              // arbitrary strings
  ListSelfieBackground    // selfie backgrounds
}
```

```
const ENHANCED_LIST_TYPES = [ListFace, ListSelfieBackground]
```

Face and SelfieBackground are designated “Enhanced” list types. face lists require a BiometricBadge component. the face list creation dialog from `AddListModal.tsx`:

```
const MAX_FACE_LIST_ITEMS_EXPIRE_AFTER_YEARS = 3
const MAX_FACE_LIST_ITEMS_EXPIRE_AFTER_DAYS = 1095

// subtitle text in the UI:
"Item state will change to Inactive and biometric data will be erased (max 3 years)"
"Item data will be erased"
```

from `AsyncSelfie.tsx`, dashboard operators can add selfies to face lists directly from verification results:

```
// "Add to Face List" dropdown
// creates ListItemFace via API with face-photo-upload
// duplicate detection: 409 "blob already exists as an active item in list"
// source tracking: sourceToken links face to original inquiry
// feature-flagged: featureListsFacesEnabled
```

operators build facial databases, selfies from verifications get added, incoming verifications get matched against them, and it’s supposedly a 3-year max retention with automatic deletion.

0x0C - the OpenAI integration: what about them?

source files:

- `dashboard/hooks/useAgentConversationStream.ts`
- `lib/constants/externalIntegrationVendors.ts`

the api.openai.com CSP entry exists because Persona built an AI copilot feature (“AskAI”) for dashboard operators.

```
// OpenAI streaming events - standard SSE format
OpenAIEventTypes:
  RESPONSE_CREATED     = 'response.created'
  OUTPUT_TEXT_DELTA     = 'response.output_text.delta'
  OUTPUT_TEXT_DONE      = 'response.output_text.done'
  RESPONSE_COMPLETED   = 'response.completed'
  RESPONSE_FAILED      = 'response.failed'
```

```
// TODO: "Remove this frontend-only cancellation logic once OpenAI
//        fixes their cancel response API"
```

OpenAI is listed as `ExternalIntegrationProductivityOpenAi`, same category as Slack and Zendesk. it seems to be a chat assistant for operators and not a surveillance data feed.

WHAT IS NOTABLE: this AI copilot runs on the same government platform that handles SARs, facial biometrics, and watchlist screening. government operators using AI chat assistance while reviewing suspicious activity reports and facial recognition matches.

the code doesn't show PII flowing to OpenAI but honestly the questions about what context the copilot has access to are worth asking.

the vendor ecosystem from `externalIntegrationVendors.ts`:

```
Chainalysis  - crypto address screening
Equifax      - credit/identity data
SentiLink    - synthetic identity fraud detection
Middesk      - business verification
Kyckr        - business registry lookups
TRM          - crypto compliance/investigation
MX           - financial data aggregation
OpenAI       - AI copilot (productivity)
Salesforce   - CRM
HubSpot      - CRM
Slack        - notifications
Zendesk      - support
```

we didn't see Palantir, Clearview, or NEC. no surveillance vendors at all. the ecosystem seems to be KYC/AML compliance and not law enforcement surveillance.

however, there's no way for us to prove that they don't have access to all of that data anyway. we can only assume that they don't have access to all of that data. but if you want my two cents, they probably do.

0x0D - watchlist screening: sanctions, PEP, adverse media

source files:

- `dashboard/models/report-template-version.ts`
- `dashboard/forms/FormReportTemplate/WatchlistListsSelect/*`

from `report-template-version.ts`, the screening configuration:

```
// watchlist screening options
'sanctions-enabled': boolean
```

```
'warnings-enabled': boolean
'fitness-probity-enabled': boolean
'recurring-enabled': boolean       // re-screen on schedule
'interval-seconds': number          // how often

// match tuning
'match-hypocorisms': boolean       // nickname matching
'match-abbreviations': boolean
'match-transliterations': boolean  // character set conversion
'match-birthplace': boolean
'match-citizenship': boolean
'match-nationality': boolean
'match-residency': boolean
'birthdate-match-type': 'day_exact' | 'year_exact' | 'plus_or_minus_one_year'
'minimum-characters-for-fuzziness': number

// match presets
enum MatchPreset { EXACT, STRICT, TOLERANT, CUSTOM }
enum MatchFuzzinessLevel { LOW, MEDIUM, HIGH, NONE }
```

custom FinCEN screening lists:

```
enum CustomListFileType {
  FincenPerson = 'fincen_person',       // FinCEN person list format
  FincenBusiness = 'fincen_business',   // FinCEN business list format
  CustomScreening = 'custom_screening', // custom format
}

// operators upload FinCEN-formatted lists
// run-policy: 'on_list_upload' | 'always' | 'manual'
// screens entire user base against them
```

PEP screening configuration:

```
'enabled-classes': PoliticallyExposedPersonClass[]
  // pep-class-1: heads of state, senior politicians
  // pep-class-2: family members
  // pep-class-3: close associates
  // pep-class-4: extended connections
'country-codes': string[]     // filter by country
'recurring-enabled': boolean  // ongoing monitoring
```

adverse media screening:

```
'search-criteria': string[]
'article-age-maximum-days': number
'article-language-allowlist': string[]
'person-age-when-article-published-minimum': number
```

```
  'only-articles-with-published-dates': boolean
  'birthdate-match-type': 'day-exact' | 'year-exact'
```

there is also business adverse media, business watchlist, and crypto address watchlist screening, but they’re all as separate report template types with their own recurring schedules.

0x0E - PEP facial recognition: comparing your selfie to world leaders

so you uploaded a selfie to use a chatbot. congratulations!!! it’s now being compared against a database of every politician, head of state, and their extended family tree on earth. similarity scored. low, medium, high. the machine looked at your face and asked itself: “does this person resemble the deputy finance minister of moldova?” and it answered. and it wrote the answer down.

we found this and had to read it three times before we believed the code was real. couldn’t stop laughing.

anyway…

source files (from source-map extraction):

- `PoliticallyExposedPersonV2EntityMatchDetails.tsx` (932 lines)
- `PoliticallyExposedPersonPhotoComparison.tsx` (372 lines)

the PEP V2 entity match view shows a full dossier for each match:

- **profile**: full name with aliases, match highlighting (exact vs fuzzy), birthdates, sex, nationality with flag icons
- **portrait similarity**: side-by-side comparison of your selfie vs reference photos. similarity level: **Low | Medium | High**. reference images sourced from Wikidata. thumbnail carousel for multiple photos. tooltip: “*Portrait similarity shows whether the end user’s portrait resembles publicly available photos.*”
- **positions**: political positions with duration and PEP class, sortable
- **candidacies**: election candidacies with election period
- **relationships**: associates with name, relation type, PEP class - can trigger related PEP reports
- **dismiss match**: IgnoreHitsModal for false positives

the photo comparison interface:

```
interface PoliticallyExposedPersonImage {
  'external-url'?: string      // reference photo from database
  'internal-url'?: string      // YOUR photo
  'similarity-level'?: 'low' | 'medium' | 'high'
}
```

there's three levels: low, medium, high. your face compared to every political figure in their database. this isn't name matching but FACE matching. with a similarity score on each comparison.

i mean… kind of makes sense. it's not like most big political figures hold various fake identities… right…

the V2 system has a known bug. source code TODO: *"Clean up this type. There is not parity between 2.0 and 1.0"*, so they're running two parallel PEP screening systems with known incompatibilities on a platform that decides whether people get reported to FinCEN… great

---

0x0F - chainalysis: crypto address surveillance

source files:

- `dashboard/components/ReportResult/ChainalysisAddressScreening.tsx` (217 lines)

the chainalysis integration screens cryptocurrency addresses with:

```
risk assessment:
  risk level: severe | high | medium | low
  risk reason: text explanation
  (severe/high = red alert icon)

cluster analysis:
  cluster name (which entity controls the address)
  cluster category

exposures:
  per-category dollar amounts
  displayed as "$" + rounded value

address identifications:
  category and description per identification
  expandable accordion for each
```

there's also a native crypto address watchlist system layered on top. operators add addresses to a watchlist, set an `interval-seconds` value, and the platform re-screens them on a cron.

this isn’t a one-shot lookup but a persistent monitor. your wallet goes on the list once and gets polled indefinitely against chainalysis’s cluster graph. every time the risk profile of an upstream cluster changes (say a mixer gets flagged, or an exchange gets sanctioned) every address downstream gets re-evaluated automatically. the recursion is quiet, so you’d never know it was happening. the only evidence is a `recurring-enabled: true` in a report template config that nobody outside this codebase was ever supposed to see…

0x10 - the full verification pipeline: 269 checks

source file: `lib/verificationCheck.ts`

the CheckName enum contains 269 individual verification checks across 14 check types. some highlights:

selfie checks (23):

```
SelfieIdComparison              - face vs ID photo
SelfieAccountComparison         - face vs existing account
SelfieLivenessDetection         - spoof detection
SelfiePublicFigureDetection     - do you LOOK LIKE someone famous?
SelfieSuspiciousEntityDetection - you look "suspicious." literally.
SelfieExperimentalModelDetection - EXPERIMENTAL ML models on your face
SelfieRepeatDetection           - duplicate selfie detection
SelfieSimilarBackgroundDetection - same background as another user
SelfieAgeComparison             - estimated age from face
SelfieAgeInconsistencyDetection - age doesn't match ID
SelfieFaceCoveringDetection     - wearing a mask? flagged.
SelfieGlassesDetection          - glasses? noted.
SelfiePoseRepeatDetection       - same pose as last time?
```

`SelfieSuspiciousEntityDetection`. what makes a face “suspicious”? the code doesn’t say. the users aren’t told.

government ID checks (43): including AAMVA database lookup (US driver’s license database), physical tamper detection, MRZ detection, electronic replica detection, NFC chip reading with PKI validation, public figure detection, Real ID detection.

database checks (27): including deceased detection (SSA death master file), social security number comparison, phone carrier checks, SERPRO (Brazil) face comparison, Aadhaar (India) database checks, TIN validation.

document checks (29): including JPEG original image detection, PDF editor detection, PDF annotation detection, synthetic content detection, digital text modification detection.

business checks: including AI identity comparison, website backlink detection, domain age check, terms of service legitimacy detection.

269 checks. for wanting to use a chatbot in 2026.

0x11 - the architecture

```
┌─────────────────────────────────────────────────────────┐
│                    the user                             │
│  (signs up for OpenAI, wants to use GPT-5)              │
└─────────────────┬───────────────────────────────────────┘
                  │
                  │ "verify your identity"
                  │
┌─────────────────▼───────────────────────────────────────┐
│  inquiry.withpersona.com                                │
│  Persona verification flow                              │
│  - government ID scan (Microblink)                      │
│  - selfie capture + LIVENESS DETECTION                  │
│  - video capture                                        │
│  - PUBLIC FIGURE FACIAL MATCHING                        │
│  - device fingerprint (FingerprintJS)                   │
│  - browser/network signals                              │
└─────────────────┬───────────────────────────────────────┘
                  │
                  │ complete identity dossier
                  │ (ID photos, selfie, video, PII, scores)
                  │
┌─────────────────▼───────────────────────────────────────┐
│  openai-watchlistdb.withpersona.com                     │
│  34.49.93.177 (dedicated GCP)                           │
│  Envoy proxy + internal service mesh                    │
│                                                         │
│  screens against:                                       │
│  - OFAC SDN list (US sanctions)                         │
│  - 200+ global sanctions/warning lists                  │
│  - PEP classes 1-4 (with FACIAL SIMILARITY scoring)     │
│  - adverse media (terrorism to cybercrime)              │
│  - crypto address watchlists (Chainalysis, TRM Labs)    │
│  - custom FinCEN screening lists                        │
│  - fitness & probity lists                              │
└─────────────────┬───────────────────────────────────────┘
                  │
                  │ result: approved / flagged / denied
                  │
┌─────────────────▼───────────────────────────────────────┐
│  OpenAI                                                 │
│  - grants or denies access                              │
│  - no explanation provided                              │
```

```
  │ - no appeal mechanism                                 │
  │ - data retained (1 year? 3 years? permanently?)       │
  └───────────────────────────────────────────────────────┘


meanwhile, on the government side:


  ┌───────────────────────────────────────────────────────┐
  │ withpersona-gov.com (FedRAMP Authorized)              │
  │ 34.27.15.233 (dedicated GCP, us-central1)             │
  │                                                       │
  │ SAME CODEBASE. same company. same data model.         │
  │                                                       │
  │ proven in source code:                                │
  │ - files SARs directly to FinCEN                       │
  │ - files STRs directly to FINTRAC (Canada)             │
  │ - STRs tagged with intelligence program codenames     │
  │ - biometric face databases (3-year retention)         │
  │ - 13 types of tracking lists                          │
  │ - PEP facial recognition with similarity scoring      │
  │ - 269 verification checks                             │
  │ - Chainalysis crypto screening                        │
  │ - custom FinCEN screening list uploads                │
  │ - OpenAI-powered AI copilot for operators             │
  └───────────────────────────────────────────────────────┘
```

the same company that takes your passport photo when you sign up for ChatGPT also operates a government platform that files Suspicious Activity Reports with FinCEN and tags them with intelligence program codenames. same codebase. confirmed by matching git commit hashes across deployments.

0x12 - the legal questions

UKRAINE IS NOT OFAC-SANCTIONED.

the OFAC Ukraine/Russia sanctions program targets specific individuals and entities connected to Russia’s occupation of Crimea. it does not sanction Ukraine as a country. yet OpenAI blocks Ukraine alongside Afghanistan, Belarus, Iran, North Korea, Russia, Syria, and Venezuela. a country being actively invaded, blocked from AI tools, not because of any legal requirement but because of a policy choice.

BIOMETRIC DATA RETENTION.

OpenAI’s disclosures reference biometric data stored “up to a year.” the source code shows face list retention capped at 3 years. government IDs retained “permanently” per Persona’s practices. which is it?

BIPA EXPOSURE.

the Illinois Biometric Information Privacy Act requires informed written consent BEFORE collection of biometric data, disclosure of purpose and storage length, and a publicly available retention schedule. OpenAI collects facial biometrics through Persona. with "millions" of monthly screenings, the statutory damages exposure is significant ($1,000 per negligent / $5,000 per willful violation).

NO TRANSPARENCY, NO RECOURSE.

community reports document: users passing verification then being locked out with no reason. "you can't try again; that's our policy." no human support. no appeal. no disclosure of evaluation criteria.

you gave them your passport. your face. your address. they said no. they won't tell you why. and they kept your data.

0x13 - what the code does NOT show

transparency matters. the source code does NOT contain:

- **no ICE integration.** zero references to ICE, immigration enforcement, deportation, border patrol, or any immigration agency in 2,456 source files. however, with everything mentioned earlier, we assume that it is likely that they are somehow getting access to your data.
- **no Fivecast ONYX connection.** zero references to Fivecast or ONYX-as-a-product. the subdomain match is real. the code connection is not proven (obviously), but this seems like too much of a coincidence to us.
- **no surveillance vendors.** no direct Palantir, Clearview, NEC, Babel Street, ShadowDragon, Voyager Labs, Cellebrite. the vendor ecosystem is KYC/AML, even though shareholder data and other information could heavily impact this statement.
- **no bidirectional OpenAI data pipeline.** OpenAI integration = AI chat copilot for operators. categorized as productivity alongside Slack and Zendesk.
- **no law enforcement features.** no warrant management, no subpoena processing, no evidence chain of custody, no criminal investigation workflows but the data CAN be sent. which means that the data can probably be used for law enforcement purposes.

the platform is a financial compliance system used by government agencies. that's concerning enough without making it something it isn't. the SAR filing to FinCEN, the biometric face databases, the PEP facial recognition, the 269 verification checks, the intelligence program tags on STR filings… these are real and confirmed in source code, and raise serious questions on their own.

obviously, the source code is not enough to determine WHO the data is being sent to. we cannot and did not hack into their platform.

0x14 - questions that deserve answers

1. what was OpenAI screening against in november 2023, 18 months before disclosing any identity verification requirements?

2. does “watchlistdb” imply a proprietary watchlist beyond OFAC/SDN/PEP? what criteria determine inclusion?

3. which federal agencies use the withpersona-gov.com platform? the code is agency-agnostic.

4. what defines a “suspicious entity” in `SelfieSuspiciousEntityDetection`? what facial characteristics trigger this flag?

5. what do the experimental model detection checks (`SelfieExperimentalModelDetection`, `IdExperimentalModelDetection`) do? unnamed ML models running on live biometric data.

6. what is the actual biometric retention period? OpenAI says “up to a year.” the code says 3 years max. government IDs retained “permanently.” which is it?

7. has a BIPA compliance assessment been performed for Illinois residents?

8. why is Ukraine blocked alongside OFAC-sanctioned countries when Ukraine itself is not subject to US sanctions?

9. what happens to the data of users who are screened and denied? retained for how long? can law enforcement access it?

0. why does a government compliance platform need an AI copilot that talks to OpenAI? what context does it have access to?

1. what is the relationship between Persona’s “onyx” deployment and Fivecast ONYX, ICE’s $4.2M surveillance tool?

2. were users informed that their selfie undergoes public figure facial matching - that the platform checks whether your face resembles a known politician?

3. how did 53 MB of unprotected source maps end up on a FedRAMP-authorized government endpoint? was this reviewed in the security assessment?

4. who authorized liveness/spoof detection that assigns “High Risk” labels recommending automatic rejection? what is the false positive rate?

5. why does a FedRAMP platform include FINTRAC filing? which agencies have cross-border filing capability?

6. the source code admits to “depending on obfuscation” for AES-256-GCM encryption keys. did FedRAMP assessors review this?

7. the PEP screening system has known V1/V2 incompatibilities. how many false positives has this caused?

8. are users informed that 269 distinct verification checks are performed - including SSN death record matching, phone carrier queries, and PDF metadata analysis?

0x15 - infrastructure reference

```
┌──────────────────────────────────────────────────────────────┐
│ PRODUCTION - OpenAI watchlist screening                      │
├──────────────────────────────────────────────────────────────┤
│ openai-watchlistdb.withpersona.com         34.49.93.177      │
│ openai-watchlistdb-testing.withpersona.com 34.49.93.177      │
│ provider: GCP (AS396982) | envoy proxy | port 443 only       │
│ cert: Google Trust Services WR3 | 90-day rotation            │
│ operational since: 2023-11-16                                │
└──────────────────────────────────────────────────────────────┘

┌──────────────────────────────────────────────────────────────┐
│ STAGING - watchlist screening                                │
├──────────────────────────────────────────────────────────────┤
│ staging-watchlistdb.withpersona-staging.com  34.49.208.72    │
│ provider: GCP | envoy proxy | port 443 only                  │
│ cert SAN includes: stateless-test.withpersona-staging.com    │
└──────────────────────────────────────────────────────────────┘

┌──────────────────────────────────────────────────────────────┐
│ GOVERNMENT - withpersona-gov.com (FedRAMP Authorized)        │
├──────────────────────────────────────────────────────────────┤
│ withpersona-gov.com / www          34.27.15.233  (GCP)       │
│ app.withpersona-gov.com            34.27.15.233  (GCP)       │
│ admin.withpersona-gov.com          34.27.15.233  (GCP)       │
│ inquiry.withpersona-gov.com        34.27.15.233  (GCP)       │
│ login-gov.withpersona-gov.com      3.15.167.135  (AWS)       │
│   └─ Okta tenant: personaforgov-admin.okta.com               │
│ app.trust.withpersona-gov.com      104.21.47.3   (CF)        │
│ trust.withpersona-gov.com          104.21.47.3   (CF)        │
│ cert: *.withpersona-gov.com wildcard | GTS WR3               │
│ FedRAMP Authorized: 2025-10-07 (Low Impact)                  │
│ FedRAMP Ready: Moderate Impact                               │
└──────────────────────────────────────────────────────────────┘

┌──────────────────────────────────────────────────────────────┐
│ ONYX - government deployment (purpose unknown)               │
├──────────────────────────────────────────────────────────────┤
│ onyx.withpersona-gov.com               34.10.180.174 (GCP)   │
│ app.onyx.withpersona-gov.com           34.10.180.174 (GCP)   │
│ admin.onyx.withpersona-gov.com         34.10.180.174 (GCP)   │
│ inquiry.onyx.withpersona-gov.com       34.10.180.174 (GCP)   │
│ cert: *.onyx.withpersona-gov.com wildcard │ GTS WR3          │
│ server: istio-envoy                                          │
│ k8s namespace: persona-onyx                                  │
│ k8s pods: web-5785f4b8f-jgm7m, web-5785f4b8f-qjg8f           │
│ k8s service: web.persona-onyx.svc.cluster.local:3000         │
│ git commit: 8d454ac0dc48b2f4ae7addefa22e746079c30089         │
│ first cert: 2026-02-04 (12 DAYS OLD)                         │
│ CSP includes: api.openai.com                                 │
│ dashboard loads: faceapi.js (facial recognition)             │
│ SOURCE MAPS: 53 MB unprotected on /vite-dev/ path            │
│ admin login: Okta-only SSO (internal staff)                  │
│ server-timing: Flipper feature flags leaked                  │
│ security.txt: EXPIRED 2025-11-01 (Bugcrowd program)          │
└──────────────────────────────────────────────────────────────┘


┌──────────────────────────────────────────────────────────────┐
│ TRUST PORTAL - FedRAMP Private Backend API                   │
├──────────────────────────────────────────────────────────────┤
│ app.trust.withpersona-gov.com      104.21.47.3    (CF)       │
│ trust.withpersona-gov.com          104.21.47.3    (CF)       │
│ CF-Access-Domain: api.trust.withpersona-gov.com              │
│ app name: "fedramp-private-backend-api"                      │
│ Cloudflare Access gate                                       │
└──────────────────────────────────────────────────────────────┘
```

0x16 - methodology

all findings obtained through passive reconnaissance using publicly available tools and data sources. no systems were accessed, no credentials were used, no vulnerabilities were exploited.

```
tools used:
  - Shodan (shodan.io)
  - crt.sh (certificate transparency)
  - DNS resolution
  - HTTP/HTTPS requests to public endpoints
  - SSL/TLS certificate inspection
  - public API documentation review
  - JavaScript source map analysis (publicly served)
  - static analysis of extracted TypeScript source

what we did NOT do:
  - no authentication attempts
  - no vulnerability scanning
  - no exploitation of any kind
  - no brute-forcing
  - no modification of any data
```

every source code finding is a direct code reference. file paths, function names, enum values, string literals - all verifiable against the source. infrastructure findings are from Shodan, CT logs, and HTTP headers.

0x17 - epilogue

start with a Shodan search. find an IP. read a certificate. follow the DNS. parse a header. download 53 megabytes of source code from a government endpoint that forgot to lock the door. and suddenly you’re reading the architecture of a system that decides who gets to use AI and who gets reported to FinCEN.

the source code reveals a platform that:

- files Suspicious Activity Reports directly with FinCEN
- files Suspicious Transaction Reports with FINTRAC
- tags STRs with intelligence program codenames (Project SHADOW, Project LEGION…)
- maintains biometric face databases with 3-year retention
- runs 269 distinct verification checks against every user
- compares your selfie to political figures with facial similarity scoring
- flags you as a “suspicious entity” based on your face alone
- classifies selfie spoof risk with hardcoded rejection thresholds
- screens against 14 categories of adverse media from terrorism to espionage
- lets operators upload custom FinCEN screening lists and run them against everyone
- continuously re-screens on configurable intervals
- tracks you across 13 types of lists from browser fingerprints to geolocations
- screens crypto wallets against sanctioned addresses via Chainalysis
- runs experimental unnamed ML models on your biometric data
- encrypts data with shared symmetric keys while admitting they “depend on obfuscation”
- runs two parallel PEP screening systems with known incompatibilities

and the company that runs all of this is the same one that takes your passport photo when you sign up for ChatGPT. same codebase. same platform. different deployment. same facial recognition. same screening algorithms. same data model.

is there a direct pipeline between OpenAI's millions of monthly screenings and the government SAR filing system? the code doesn't prove it. but the code does prove that Persona operates both systems, that both run the same software, and that both are live right now.

2,456 source files. 269 verification checks. 13 list types. 7 intelligence program codenames. 3-year biometric retention. 2 parallel PEP systems with bugs. 1 FedRAMP authorization. and an ONYX deployment that appeared 12 days ago whose purpose nobody will explain.

the information is the moral argument. you're reading it.

0x18 - betrayal

the more sad part about this is that the people working on this surveillance are or were more than likely your yearmates. perhaps you were in the same 101 class, or encountered each other in the hallways or an internship event. these same people are now working for surveillance, for helping the current government, and for future ones to come (because, lets face it, fascist surveillance is not exclusive to the current administration or the US).

so in light of transparency to the people, friends, partners, and family, that these individuals have ever come into contact with, here is a full list of all verifiable associations compared against every single comment in the 2456 source files.

this isn't a sorted list, there is no malice in whatever order these individuals are listed. this is purely for transparency.

notable and in-common universities:

- University of Waterloo
- Brown
- University of Maryland
- UPenn
- Berkeley
- Harvey Mudd
- Rochester Institute of Technology
- University of Central Florida
- Carnegie Mellon University
- Caltech

- UC San Diego
- Stanford
- MIT
- Cornell
- CSU LA
- University of Southern California
- University of British Columbia
- Barnard College

ADDENDUM — february 18, 2026

this used to host a list of people who have worked on this, but since the internet can't behave after 46 years and feels the need to perform some vigilante justice against people they dont know at all—when the intended audience was family members and friends, you can enjoy finding that info yourself now

sources

#	source
1	Shodan: 34.49.93.177
2	Shodan: 34.27.15.233 (gov)
3	CT logs: openai-watchlistdb
4	CT logs: withpersona-gov.com
5	Persona case study: OpenAI
6	Persona FedRAMP announcement
7	Persona Government ID Verification API
8	OFAC Ukraine/Russia sanctions
9	OpenAI API Supported Countries
10	OpenAI Organization Verification
11	Community: broken Persona verification
12	Illinois BIPA (740 ILCS 14/)
13	CT logs: onyx.withpersona-gov.com
14	EFF: ICE surveillance shopping spree
15	Contract: ICE Fivecast ONYX ($4.2M)
16	Contract: CBP Fivecast ONYX licenses
17	Fivecast ONYX platform

#	source
18	Source map: dashboard-BgoO_aj5.js.map (18.7 MB)
19	Source map: faceapi-CCSM7NPL.js.map (2.8 MB)
20	Shodan: 34.10.180.174 (ONYX)
21	Persona Carahsoft gov partnership
22	FINTRAC
23	@vladmandic/face-api
24	NIST SP 800-53

source code references - all from publicly served source maps at `app.onyx.withpersona-gov.com/vite-dev/` and website download at `withpersona-gov.com`:

- `lib/verificationCheck.ts` - 269 CheckName enum values
- `dashboard/models/filing.ts` - FincenStatus, FintracStatus enums
- `dashboard/views/DashboardSARShowView/` - SAR filing module
- `dashboard/views/DashboardFilingShowView/components/STRFormSchema.tsx` - FINTRAC STR schema
- `dashboard/models/report-template-version.ts` - screening configuration
- `dashboard/components/ReportResult/ChainalysisAddressScreening.tsx` - crypto screening
- `dashboard/components/ReportResult/PoliticallyExposedPersonV2EntityMatchDetails.tsx` - PEP dossier
- `dashboard/components/ReportResult/PoliticallyExposedPersonPhotoComparison.tsx` - facial comparison
- `dashboard/components-lib/AsyncSelfie/AsyncSelfie.tsx` - face list management
- `dashboard/views/DashboardListsView/AddListModal.tsx` - list creation (3-year max)
- `dashboard/lib/constants/list.ts` - 13 list types
- `lib/constants/externalIntegrationVendors.ts` - vendor ecosystem
- `dashboard/hooks/useAgentConversationStream.ts` - OpenAI copilot
- `lib/api-permissions.ts` - external API permissions

---

and as always, the information wants to be free. we didn’t break anything. we didn’t bypass anything. we queried URLs, pressed buttons, and read what came back. if that’s enough to expose the architecture of a global surveillance platform… maybe the problem isn’t us.

stay curious. stay paranoid. rotate your keys. read your source maps. and if someone asks you to take a selfie to prove you’re human, ask yourself who’s on the other side of that camera, and what list you just landed on.

knowledge is the only real currency. everything else is just access control.

hack the planet~

// end of transmission //